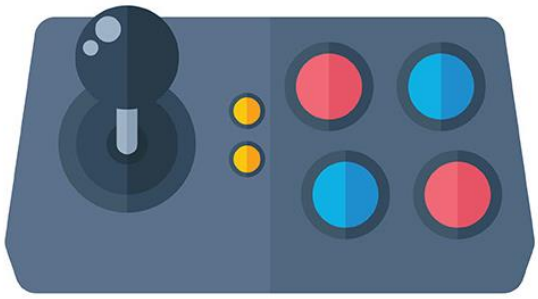
ARCADE



Julien Augugliaro / Eloïse Richard / Marius Contoli

L'Arcade est une simulation de borne d'arcade. Il s'agit d'un programme qui permet à l'utilisateur de choisir un jeu et d'avoir accès à un historique des meilleurs scores. Pour pouvoir gérer le changement de jeux et d'interface graphiques, les librairies doivent être implémentées comme des librairies dynamiques et chargées durant l'exécution du jeu.

Le but de cette documentation est de permettre à tout développeur d'ajouter une nouvelle librairie graphique/de jeu.

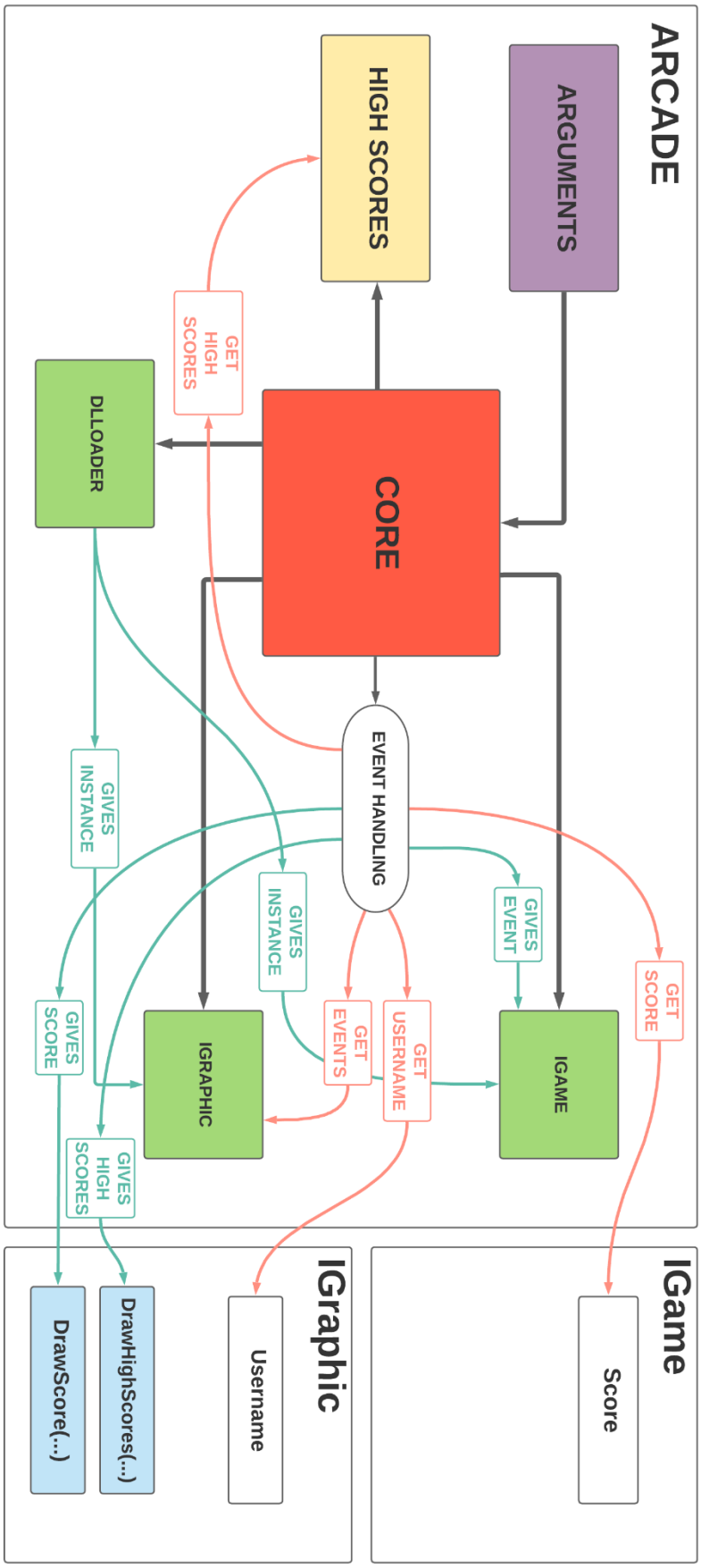
Table des matières :

I) Comment installer l'Arcade ?

1. Cloner le repo.

2. Installer les librairies utilisées par le projet :
   SFML : https://www.sfml-dev.org/download-fr.php
   SDL2 : https://www.libsdl.org/download-2.0.php
   NCurses : https://www.cyberciti.biz/faq/linux-install-ncurses-library-headers-on-debian-ubuntu-centos-fedora/

II) Architecture



III) Standards et interfaces du projet

a) Games

```
typedef enum {
    NO_GAME,
    NIBBLER,
    PACMAN,
    GAME_OVER,
} game_e;
```

game_e permet de définir le jeu en cours d'utilisation ou l'état de l'Arcade.

b) Events

```
typedef enum {
    NONE,
    UP,
    DOWN,
    LEFT,
    RIGHT,
    PREV_GAME,
    NEXT_GAME,
    RESET_GAME,
    GO_MENU,
    CONFIRM_NAME,
    PREV_GRAPH,
    NEXT_GRAPH,
    QUIT_GAME
} evtKey;
```

evtKey permet de définir tous les événements que le Core peut gérer.

c) Clocks

Les clocks sont implémentées directement dans les librairies de jeu, et permettent la gestion du rafraichissement et du delta time.

La classe AClock agit comme un wrapper autour des fonctions de la librairie ctime pour permettre son utilisation tout en respectant les normes modernes du C++.

AClock.hpp

```
class AClock {
    public:
        AClock();
        ~AClock();
        double GetElapsedTime() const;
        void Restart();

    protected:
    private:
        clock_t _clock;

};
```

AClock.cpp

```
AClock::AClock()
{
    _clock = clock();
}

AClock::~AClock()
{

}

double AClock::GetElapsedTime() const
{
    return ((double)(clock() - _clock) / CLOCKS_PER_SEC);
}

void AClock::Restart()
{
    _clock = clock();
}
```

IV) Interface IGame

L'interface IGame étant minimaliste, permet d'implémenter facilement une librairie de jeu.

```
class IGame {
    public:
        virtual ~IGame() = default;

        virtual std::vector<std::string> GetMap() const = 0;
        virtual int GetScore() const = 0;
        virtual bool IsGameOver() const = 0;
        virtual void Update(evtKey key) = 0;
        virtual game_e GetGameName() const = 0;
        virtual AClock &GetDeltaTime() = 0;

};
```

La fonction GetMap() permet de récupérer la map du jeu et de tous les éléments la composant.

La fonction GetScore() permet de récupérer le score du jeu tournant actuellement.

La fonction IsGameOver() permet de savoir si le jeu doit passer sur l'affichage de l'écran game over.

La fonction Update() permet de mettre à jour l'état du jeu.

La fonction GetGameName() permet de récupérer l'identifiant du jeu actuel.

La fonction GetDeltaTime() permet de récupérer le delta time.

V) Interface IGraphic

L'interface IGraphic permet d'implémenter n'importe quelle librairie graphique facilement.

```
class IGraphic {
    public:
        virtual ~IGraphic() = default;

        virtual void Initialize() = 0;
        virtual void DrawMap(std::vector<std::string> map) = 0;
        virtual void DrawScore(int score) = 0;
        virtual void DrawHighScores(std::vector<std::pair<std::string,
 std::string>> &pacman_list, std::vector<std::pair<std::string, std::s
tring>> &nibbler_list) = 0;
        virtual void Display(AClock &delta) = 0;
        virtual void DisplayMenu() = 0;
        virtual void DisplayGameOver() = 0;
        virtual void Destroy() = 0;
        virtual void Clear() = 0;
        virtual void Close() = 0;
        virtual evtKey GetEventKey() const = 0;
        virtual std::string GetUsername() const = 0;
        virtual evtKey InputGameOverName() = 0;

};
```

La fonction Initialize() initialise les variables permettant le bon fonctionnement de la librairie.

La fonction DrawMap() permet l'affichage de la map sur la fenêtre.

La fonction DrawScore() permet l'affichage du score sur la fenêtre.

La fonction DrawHighScores() permet l'affichage des highscores sur la fenêtre.

La fonction Display() est la fonction principale d'affichage du jeu.

La fonction DisplayMenu() est la fonction principale d'affichage du menu.

La fonction DisplayGameOver() est la fonction principale d'affichage de l'ecran de game over.

La fonction Destroy() permet la désallocation des ressources utilisées par la librairie graphique, elle est appelée par le destructeur.

La fonction Clear() permet de nettoyer tous les éléments affichés à l'écran.

La fonction Close() permet de fermer proprement la fenêtre liée à la librairie graphique.

La fonction GetEventKey() permet de récuperer l'event associé à la pression d'une touche du clavier.

La fonction GetUsername() permet de récupérer le nom du joueur stocké dans le jeu.

La fonction InputGameOverName() permet de gérer les events associés à l'insertion du nom du joueur dans l'écran de game over.